

Python for Data Science

1. Introduction to Python Programming



Contents

- 1.1 Variables, Expressions and Assignment Statements**
- 1.2 Strings**
- 1.3 Getting Input from the User**

1

Variables, Expressions and Assignments

Variables Expressions

Python **expressions** also may contain **variables**, which store **values** for later use in your code.

```
In [1]: 45 + 72
```

```
Out[1]: 117
```

```
In [2]: x = 7
```

Assignments

Snippet [2] is a **statement**.

The preceding statement creates **x** and uses the **assignment symbol** (=) to give **x** a value. The entire statement is an **assignment statement**.

Variable Names

A variable name, such as **x**, is an **identifier**.

Each identifier may consist of letters, digits and underscores but may not begin with a digit.

Python is **case sensitive**.

1

Variables, Expressions and Assignments

Types

Each value has a **type** that indicates the kind of data the value represents.

```
In [7]: type(x)
```

```
Out[7]: int
```

```
In [8]: type(10.5)
```

```
Out[8]: float
```

type()

Python's **type** built-in function determine a value's type.

A **function** performs a task when you **call** it by writing its name, followed by parentheses.

The parentheses contain the function's **argument** – the data that the **type** function needs to perform its task.

1

Variables, Expressions and Assignments

Arithmetic

The following table summarizes the **arithmetic operators**.

Python operation	Arithmetic operator	Algebraic expression	Python expression
Addition	+	$f + 7$	<code>f + 7</code>
Subtraction	-	$p - c$	<code>p - c</code>
Multiplication	*	$b \cdot m$	<code>b * m</code>
Exponentiation	**	x^y	<code>x ** y</code>
True division	/	x/y or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
Floor division	//	$\lfloor x/y \rfloor$ or $\left\lfloor \frac{x}{y} \right\rfloor$ or $\lfloor x \div y \rfloor$	<code>x // y</code>
Remainder (modulo)	%	$r \bmod s$	<code>r % s</code>

1

Variables, Expressions and Assignments

Exceptions

Dividing by zero with / and // is not allowed and results in an **exception**.

```
In [10]: 123 / 0
```

```
ZeroDivisionError
```

```
Traceback (most recent call last)
```

```
<ipython-input-10-cd759d3fcf39> in <module>()
```

```
----> 1 123 / 0
```

```
ZeroDivisionError: division by zero
```

Tracebacks

Python reports an exception with a **traceback**.

This traceback indicates that an exception of type `ZeroDivisionError` occurred.

[Expressions — Python 3.10.5 documentation](#)

2

Strings

`print()`

The built-in `print` function displays its argument as a line of text.

The argument `'Welcome to Python!'` is a **string** – a sequence of characters enclosed in single quotes.

You also may enclose a string in double quotes.

```
In [1]: print('Welcome to Python!')  
Welcome to Python!
```

```
In [2]: print("Welcome to Python!")  
Welcome to Python!
```

```
In [3]: print('Welcome', 'to', 'Python!')  
Welcome to Python!
```

Escape Sequence

When a backslash (\) appears in a string, it's known as the **escape character**.

The backslash and the character immediately following it form an **escape sequence**.

Escape sequence	Description
\n	Insert a newline character in a string. When the string is displayed, for each newline, move the screen cursor to the beginning of the next line.
\t	Insert a horizontal tab. When the string is displayed, for each tab, move the screen cursor to the next tab stop.
\\	Insert a backslash character in a string.
\"	Insert a double quote character in a string.
\'	Insert a single quote character in a string.

3

Getting Input from the User

Escape Sequence

The `input` function requests and obtains user input.

The function always returns a string.

```
In [1]: name = input("What's your name? ")  
What's your name? Paul
```

```
In [2]: name  
Out[2]: 'Paul'
```

```
In [3]: print(name)  
Paul
```

```
In [7]: value1 = input('Enter first number: ')  
Enter first number: 7
```

```
In [8]: value2 = input('Enter second number: ')  
Enter second number: 3
```

```
In [9]: value1 + value2  
Out[9]: '73'
```

3

Getting Input from the User

Getting an Integer

If you need an integer, convert the string to an integer using the `int` function.

If the string passed to `int` cannot be converted to an integer, a `ValueError` occurs.

To convert strings to floating-point numbers, use the `float` function.

```
In [10]: value = input('Enter an integer: ')\nEnter an integer: 7
```

```
In [11]: value = int(value)
```

```
In [12]: value\nOut[12]: 7
```